Amendments to the Specification

Please replace the title on the title page, on page 1 and on page 18 with the following revised title:

Method and ~~Internal~~ Pipeline Architecture for Processing Multiple Swap Requests ~~Save/restore Operation~~ to Reduce Latency

Please replace the paragraph number 4 with the following revised paragraph:

[4] Figure 1A shows a typical prior art processing window 100 used by a pipeline processor. The processing window 100 includes some number of register windows 110-117 (e.g., eight register windows reg0 – reg7). An active register window 120 is also included. The active register window 120 can be accessed by the processor. By way of example, as the pipeline progresses, the contents of register window 113 will be restored to the active register 120 so that the contents of register window 113 can be processed by the processor. However, typically the active register 120 is not empty and therefore the data in the active register 120 must be saved to the appropriate register window (e.g., register window 114) before the contents of register window 113 ~~can be~~ can be restored to the active register 120. Therefore, the contents in the active register 120 must be swapped (i.e., a save operation to register window 114 and a restore operation from register window 113).

Please replace the paragraph number 6 with the following revised paragraph:

[6] The swap requests typically operate in an acceptable manner as long as the swap requests do not occur too often. However, multiple swap requests often

occur immediately following one another. As a result, the pipeline can stall. By way of example, if a first swap request (i.e., swap A) is received, then a save A operation 120A occurs, followed by a restore A operation 120B, in the next clock cycle. A second swap request (swap B 122) is received immediately after the swap A request is received. However, two clock cycles are required to complete the save A 120A and restore A 120B operations required for the swap A request, therefore the swap B request 122 must be delayed two clock cycles before the swap B request can be acted upon in respective save B operation 122A and restore B operation 122B. As a result, the swap B request 122 has a latency of 2 (two clock cycles between when swap B request was received and when restore B operation 122B was completed). If a swap C request 124 immediately followed the swap B request 122 (i.e., during the third clock cycle), then the swap C request would have a latency of 4 (i.e. four clock cycles between when the swap C request was received and when the respective save C operation 124A and subsequent restore C operation 124B was completed). Further, for an nth subsequent swap request 126 ~~n-subsequent swap requests~~, each swap request would have a n+1 latency factor. By way of example a tenth consecutive swap request (swap J) would have a latency of 11 (i.e., 11 clock cycles would pass between when the swap J request 126 was received and when the respective save J operation 126A and subsequent restore J operation 126B was completed). The latencies cause pipeline stalls where processing is delayed for that number of clock cycles.

Please replace the paragraph number 29 with the following revised paragraph:

[29] Figure 2A is a timing diagram 200 of multiple swap requests, in accordance with one embodiment of the resent invention. Figure 2B is a flowchart diagram 250 that illustrates the method operations performed in comparing multiple subsequent

swap requests in accordance with one embodiment of the present invention. In an operation 255, a first swap request (e.g., swap A 120) is received at time T0. In operation 260, the first swap request 120 is executed at time T0. When the first swap request 120 is executed, the save A operation 120A is executed during the first clock cycle and the restore A 120B is executed during the second clock cycle (i.e., between time T1 and time T2).

Please replace the paragraph number 30 with the following revised paragraph:

[30] In an operation 265, a subsequent swap request (swap B 122) is received at time T1, at the beginning of the second clock cycle, substantially simultaneously as the restore A 120B is being executed. In an operation 270, the subsequent swap request (swap B 122) is compared to the first swap request (swap A 120). The subsequent swap request 122 can be compared to the first swap request 120 as the subsequent swap request is received in operation 265 above.

Please replace the paragraph number 31 with the following revised paragraph:

[31] If in operation 270, the subsequent swap request 122 is not swapping the same register as the first swap request 120, then the method operations continue in operation 275. If in operation 270, the subsequent swap request 122 is swapping the same register as the first swap request 120, then the method operations continue in operation 280.

Please replace the paragraph number 32 with the following revised paragraph:

[32] In an operation 275, the subsequent swap request 122 is executed at a time T1. When the second swap request 122 is executed, the save B operation 222A is executed during the second clock cycle and the restore B 222B is executed during

the third clock cycle (i.e., between time T2 and time T1). As a result, the save B operation 222A is executed substantially simultaneously as the restore A operation 120B is executed. The method operations then end.

Please replace the paragraph number 33 with the following revised paragraph:

[33] In an operation 280, a delay is inserted because both of the first swap 120 and the second swap 122 are attempting to swap the same register (e.g., i<5>). By way of example, a one clock cycle delay is inserted so that the subsequent swap 122 is executed after the first swap 120 instruction is completed such as when swap C 124 is executed (i.e., save C operation 114A and restore C operation 114B are completed between times T2 and T4) as shown in Figure 2A. The delay is inserted to avoid a "collision" at the same register because two instructions would be attempting to access the same register at the same time. It should be understood that the delay can also be more than one clock cycle. The method operations then continue as described in operation 275 above except delayed by one clock cycle. Additional registers can be needed to carryout the method operations described in Figures 2A and 2B.